

09/22/99
JCS586 U.S. PTO

Express Mail Label No.: EL443489750US

PATENT

Case Docket No.: 99 P 7817 US

Date: September 22, 1999

A

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231
Box PATENT APPLICATION

Sir:
Transmitted herewith for filing is the patent
application of:

Inventor: Liu et al.

For: AUTOMATED GENERATION OF CARD-BASED
PRESENTATION DOCUMENTS FROM ...

This application includes:

17 pages: specification and claims
20 sheets of drawings, formal/ X informal
 photographs

Also enclosed is:

 Declaration and Power of Attorney
 Information Disclosure Statement pursuant to 37 CFR 1.56.

The filing fee has been calculated as shown below:

FOR:	(Col. 1) NO. FILED	(Col. 2) NO. EXTRA RATE	OTHER THAN A SMALL ENTITY FEE
BASIC FEE	XXXXXXXX	XXXXXXXXXX	XXXXX \$760.
TOTAL CLAIMS	16 - 20 =	24	x 18.= \$0
INDEP CLAIMS	3 - 3 =	0	x 78= \$0
* MULTIPLE DEPENDENT CLAIM PRESENTED			+260.= \$0.
			TOTAL \$760.

- X Please charge my Deposit Account No. 19-2179 in the amount of \$760.00. A duplicate copy of this sheet is enclosed.
- X The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. 19-2179. A duplicate copy of this sheet is enclosed.
- X Any additional filing fees required under 37 CFR 1.16.
- X Any patent application processing fees under 37 CFR 1.17.
- X The Commissioner is hereby authorized to charge payment of the following fees during the pendency of this application or credit any overpayment to Deposit Account No. 19-2179. A duplicate copy of this sheet is enclosed.
- X Any patent application processing fees under 37 CFR 1.17.
- X Any filing fees under 37 CFR 1.16 for presentation of extra claims.

Donald B. Paschburg
Donald B. Paschburg
Registration No. 33,753

Siemens Corporation
Intellectual Property Department
186 Wood Avenue South
Iselin, NJ 08830
Tel. (732) 321-3191

JCS572 U.S. PTO
09/22/99
401383

**APPLICATION FOR LETTERS PATENT
OF THE UNITED STATES**

NAME OF INVENTORS: PEIYA LIU
39 DAVISON AVENUE
EAST BRUNSWICK, NJ 08816
UNITED STATES OF AMERICA

LIANG-HUA HSU
4 ORLY COURT
ROBBINSVILLE, NJ 08691
UNITED STATES OF AMERICA

YOUNG FRANCIS DAY
35-24 QUAIL RIDGE DRIVE
PLAINSBORO, NJ 08536
UNITED STATES OF AMERICA

VIVEK AGRAWALA

TITLE OF INVENTION: **AUTOMATED GENERATION OF CARD-BASED
PRESENTATION DOCUMENTS FROM MULTIMEDIA
DATA**

TO WHOM IT MAY CONCERN, THE FOLLOWING IS
A SPECIFICATION OF THE AFORESAID INVENTION

Automatic Generation of Card-Based Presentation

Documents from Multimedia Data

Background of the Invention

5

Field of the Invention

The present invention relates to multimedia document presentations and more particularly to a system for automatically generating platform-independent formatting object descriptions from SGML documents and non-textual media sources which can then be used to create card-based multimedia presentation documents.

Description of the Prior Art

15 Traditionally, hypermedia presentations are interactively designed and manually created with hypermedia authoring tools. Various commercial hypermedia authoring tools adopt different interactive authoring paradigms but they are limited in supporting the automated generation of card-based presentation from existing multimedia document sources in large scale. The representative presentation authoring tools are summarized as follows.

25 PowerPoint from Microsoft is based on a structure-oriented model and supports hierarchical information content authoring in a 2D layout. Many commercial word processing tools follow this authoring model. Documents are often developed in terms of hierarchical structures such as book, pages or slides, sections, subsections, etc., and WYSWIG user interfaces are provided to support structure editing and issue formatting commands interactively.

Authorware from Macromedia and IconAuthor from AimTech are based on a flowchart model and use icons representing events such as audio or video, if-then functions, branching and hyperlinks in a linear progression like flowchart control. Content editors to assign real media files and properties to each icon could be used. This model provides high-level program control and global ordering of information presentation.

Director from Macromedia, based on a time-line model, displays media sequences as tracks and specializes in synchronizing events. It can be used to create high-level multimedia presentations.

Multimedia Toolbook from Asymetrix, based on an object-oriented model with scripting capability, provides more support of complex interaction behavior. Users can interactively enter document content (in multimedia objects) and define object properties including various formatting commands and scripts for object behavior. This authoring tool allows the user to get to a lower-level control of object and system behavior with script commands.

Summary of the Invention

The present invention provides a new approach for automatic generation of card-based document presentation from multimedia data sources. An *automatic card-based presentation generation system* programmatically creates card-based presentation documents from source documents. In the prior art, the card-based document presentation is interactively created by authoring tools. However, if documents are in large scale and information content already exists in other forms (e.g., archive formats), the interactively authoring process for manually creating presentation forms from these existing documents becomes

tedious and time-consuming. This invention includes an automated system for transforming a card layout style specification into presentation specifications which at run-time, are used to generate formatting object descriptions. The formatting object descriptions can then be used to actually create card-based presentation documents.

The present invention comprises a *presentation style transformer* and a *card-based presentation generator* for generating card-based Formatting Object Descriptions (FOD) from the Card Layout Style Specification (CLSS). The presentation style transformer takes the card layout style specification (CLSS) and the *Card Display Schema* (CDS) as input and transforms them into a *Card-based Presentation Specification* (CPS). CLSS is used to declaratively specify the layout of the card-based presentation. CDS is used to specify *meta rules* for presentation resources and for variable definitions. CPS is a procedural specification of overall card-based presentation characteristics including layout, resources and presentation procedures. The card-based presentation generator takes the card-based presentation specifications and card-based document content as input to create formatting object descriptions. The generator first uses a *Presentation Construct Mapper* for translating CPS constructs into constructs required by *Card-Based DSSSL Style Specifications* and then uses a *Card-based DSSSL Processor* to create a *Card-Based Document Flow Object Tree*. After that, the generator uses an *FOD Converter* to convert the flow object tree into formatting object descriptions.

Brief Description of the Drawings

Figure 1 illustrates a block diagram of the

automatic card-based presentation generation system of the present invention.

Figure 2 illustrates the syntax of a card layout style specification.

5 Figure 3 gives an example of a card layout style specification for CardTitle.

Figure 4 illustrates the syntax of a generated card-based presentation specification.

10 Figure 5 gives an example of a card-based presentation specification for CardTitle.

Figure 6 illustrates a block diagram of a presentation style transformer for translating card layout style specifications into card-based presentation specifications based on display schema specifications.

15 Figure 7. illustrates a display schema for generating a TextbyGfxTitle resource declaration.

Figure 8 illustrates a syntax of display schema for generating resource declarations or variable definitions.

20 Figure 9 illustrates a display schema for generating content variable definitions.

Figure 10 illustrates a display schema for generating all primitive resource declarations.

Figure 11 illustrates a card-based context tree.

Figure 12 illustrates card-based context paths.

25 Figure 13 illustrates a flow chart of the content mapping rule generation process.

Figure 14 illustrates examples of generated CPS for C, CT and T nodes.

30 Figure 15 illustrates examples of generated CPS for R and A nodes.

Figure 16 illustrates a block diagram of a card-based presentation generator for translating card-based presentation specifications into structured formatting object descriptions.

Figure 17 illustrates a generated card-based DSSSL style for CardTitle.

Figure 18 illustrates some card-based DSSSL style functions used for CardTitle.

5 Figure 19 illustrates a CardTitle formatting object description.

Figure 20 illustrates a general syntax of formatting object description.

10 Detailed Description of the Invention

This patent application is related to copending U.S. patent application entitled "A Generalized System for Automatically Hyperlinking Product Documents", Attorney Docket No. 99P7818US, filed on September 22, 1999 and
15 assigned to the same assignee as the present invention.

The present invention is an automatic card-based presentation generation system that programmatically generates card-based formatting object descriptions from large SGML textual documents and non-textual media
20 sources. The formatting object descriptions can be used to automatically create card-based presentation document formats. The prior art of the presentation approach is based on interactive authoring tools for manually creating multimedia presentation. The interactive
25 approach does not support creating large-scale presentation documents for industrial applications due to required human involvement in the authoring process.

The overall card-based presentation generation system is described in Figure 1. The generation system
30 comprises presentation style transformer 12 and card-based presentation generator 14. Presentation style transformer 12 receives a card layout style specification (CLSS), examples of which are shown in Figures 2 and 3. The CLSS language is described in US Patent Application

serial number 08/984,734 filed on December 4, 1997,
entitled "Style Specifications For Systematically
Creating Card-Based Hypermedia Manuals" and assigned to
the same assignee as the present invention. Presentation
5 style transformer 12 also receives a card display schema,
an example of which is shown in Figure 8. Presentation
style transformer 12 generates a card-based presentation
specification, shown in Figures 4 and 5, as output. The
generated card-based presentation specification comprises
10 two parts: MACRO resource declarations and procedural
element mapping rules.

Card-based presentation generator 14 receives both,
the card-based presentation specification and the SGML
document content as inputs and generates platform-
15 independent formatting object descriptions, FODs, as
output. Formatting object descriptions are abstract
descriptions of formatting directives of presentation
documents. Once FODs are created, an automatic scripting
process can be used to generate a card-based document
20 presentation. The automatic scripting process is
described in US Patent Application serial number
08/986,270 filed on December 5, 1997, entitled
"Formatting Card-Based Hypermedia Documents By Automatic
Scripting" and assigned to the same assignee as the
25 present invention.

Presentation style transformer, 12 of Figure 1, is
shown in greater detail in Figure 6. The presentation
style transformer comprises two major components:
resource generator 22 and style proceduralizer 24.
30 Resource generator 22 is based on the card display schema
and creates presentation resource declarations from the
card-based layout style specification. Card presentation
resource declarations include two types of resources:
MACRO_Resource(stylename, parameters) for primitives and
35 MACRO_name(stylename, parameters) for composites. The

first argument of a MACRO resource declaration must be a
stylename. Primitive resources are used for basic
presentation object resource requirements of the target
presentation platform. Composite resources can be
5 defined in terms of primitive ones or other composite
resources.

Resource generator 22 creates MACRO resource
declarations by examining the objects inside background
list and background list of card-based style
10 specifications and the corresponding card display schema
for the objects. As an example, the generated CardTitle
MACRO resource declaration is shown in Figure 5. This
generation procedure is triggered by the TextByGfxTitle
style layout specification shown in Figure 3. The
15 corresponding display schema, shown in Figure 7, is used
to guide the generator to create the TextByGfxTitle
resource declaration with default values of their
parameters. The MACRO_TitleField specifies the needed
presentation objects and the variable \$CardTitle_RTF\$
20 whose value would be calculated by a predefined DSSSL
function RTF-TitleGen from source document content at
run-time.

The display schema syntax is shown in Figure 8. It
can be used to guide the generator for both creating
25 resource declarations in the <MACRO_Declaration> part and
variable definitions in the <ContentMapping> part of the
card-based presentation specification. Display schema
for generating a variable definition of \$CardTitle_RTF\$
is done by style proceduralizer 24, shown in Figure 9.
30 Display schema for generating all primitive resource
declarations is shown in Figure 10.

Style proceduralizer 24, in Figure 6, comprises
three components: context tree builder 26, content node
path walker 28 and content mapping rule generator 30.
35 Context tree builder 26 creates a context tree for rule

generator 30 to create content mapping rules along
content node paths in the tree. Style proceduralizer 24
translates a declarative card layout style specification
into procedural card content mapping rules. For example,
5 CardTitle element content mapping is shown in Figure 5.
The first element mapping rule procedurally specifies the
following things: Any document content with tag CardTitle
or ANYDOCX under document tag CARDX which is under tag
CARD will use an element rule with matched context tags
10 for creating formatting object descriptions. Since the
second element rule matches the context tags (Card,
CardX, CardTitle), it will be invoked for further
specifying how to create formatting object descriptions
for CardTitle. The second rule basically specifies which
15 types of formatting objects are created by using a
stylename attribute value under a different cardtype.

Context tree builder 26 takes all context
information in the card layout style specifications as
input and constructs a context tree as output by
20 examining and ordering all context tags appearing in the
context and AIU attribute values in the layout
specifications, by grouping the same context
specification into one with type differences, and by
marking context tags into the following categories: R
25 (root node), C (content node) for both being a leaf node
of context tree and the last element of some context
attribute value, T (transition node) for both being a
non-leaf node and non-content node, CT
(content/transition node) for both being a non-leaf node
30 of context tree and the last element of some context
attribute value, and A (associated node) from attribute
AIU values for the context of associated AIU in graphics.
The context tree is designed to capture content mapping
rule context for making an efficient generation process

of procedural rule mappings in CPS. As an example, a context tree is shown in Figure 11.

Content node path walker 28 takes the context tree as an input and finds all context paths as the output.

- 5 The context paths are defined to be the shortest paths from the root node to each C, CT and A nodes as shown in Figure 12.

Content mapping rule generator 30 uses categories and types of nodes to generate constructs of content mapping rules. The detailed process is shown in Figure 10 13. Examples are shown in Figures 14 and 15.

- For transition nodes, mapping rule generator 30 adds element rule constructs which simply specify how to pass rule processing to their children in the context tree.
- 15 For content nodes, the mapping rule generator adds specifications of the formatting object descriptions. For content transition nodes, it adds rule specifications for both content and transition functionality. For root and associated nodes, the mapping rule generator adds
- 20 predefined and domain-specific presentation specifications. Particularly for the root node, the rule generator adds the specifications of defining an overall card-based presentation structure. For associated nodes, the mapping rule generator adds structure-specific
- 25 association specifications. For instance, the structure-specific specifications can be related to hotspots, annotations, or animation objects associated with graphic objects.

Card-based document presentation generator (14 of Figure 1) shown in Figure 16 comprises the following components: presentation construct mapper 42, card-based DSSSL processor 44 and FOD convertor 46.

Presentation construct mapper 42 is used to map constructs in CPS into Card-based DSSSL constructs shown in Figures 17 and 18. The mapping procedure is a one-

35

pass translation from CPS to a card-based DSSSL style specification as follows. Card-based DSSSL style specification uses the ISO standard document style language, called DSSSL (Document Style Semantics and Specification Language).

(1) Map CPS ElementRule context into card-based DSSSL element context.

(2) Map CPS CaseExpr and IfExpr into card-based DSSSL case and if functions.

(3) Map CPS MACRO_Declaration into card-based predefined DSSSL functions.

(4) Map CPS ProcessGrove into card-based DSSSL Process function.

(5) Map CPS DefineVar, AddObject and AddGroup into card-based DSSSL FODfo flow object instance.

This mapping process is very efficient, since CPS is designed to be as close to card-based DSSSL style specification as possible. In particular, the card-based DSSSL style specifications are based on one universal card-based presentation flow object, called FODfo, for creating the card-based document flow object tree as shown in Figure 16. A card-based document flow object tree consists of a sequence of FODfo object instances. It is an abstract representation of card-based document formatting object descriptions. Using this single universal flow object can greatly simplify mapping of constructs from CPS to card-based DSSSL style specifications.

To create an instance of FODfo flow object, the mapper only needs to make a DSSSL function *makeFODfo* call, which takes three keyed arguments: FOtype, FOname and FOinstruction. FOinstruction is a data string that represents formatting information such as the presentation object attribute values. The key concept here is that FODfo flow object is designed in such a way

that a complete card-based formatting object description can be composed by appending a sequence of FODfo object instances. Thus, a FOD can be created by a DSSSL predefined function *sosofo-append* (specification of sequence of flow object-append) to append all FODfo instances.

Card-based DSSSL processor 44 is used to create a card-based document flow object tree. This is an abstract representation of card-based document formatting descriptions and it comprises a sequence of FODfo flow objects. Each FODfo flow object contains an SGML data string of FOD and each FODfo flow object is designed for uniformly representing card-based formatting object information.

Finally, FOD converter 46 is used to convert this abstract representation of sequences of FODfo objects into FODs, i.e., formatting object descriptions, shown in Figures 19 and 20.

The card-based presentation specification is designed for bridging the gap between the declarative card layout style specification and the procedural card-based DSSSL style specification. These specifications can be automatically generated from card layout style specifications and can be used for automatically generating card-based DSSSL style specifications.

It is not intended that this invention be limited to the hardware or software arrangement or operational procedures shown disclosed. This invention includes all of the alterations and variations thereto as encompassed within the scope of the claims as follows.

CLAIMS

1. A system for automatic generation of card-based
5 presentation documents from multimedia data comprising:
a presentation style transformer; and
a card-based presentation generator connected to
said presentation style transformer.
- 10 2. A system for automatic generation of card-based
presentation documents from multimedia data as claimed in
claim 1 wherein said presentation style transformer
comprises:
a processor for receiving a card display schema and
15 for processing said card display schema to describe meta
rules about presentation resources and content variable
definitions for a card-based presentation specification.
3. A system for automatic generation of card-based
20 presentation documents from multimedia data as claimed in
claim 1 wherein said presentation style transformer
comprises:
a resource generator; and
a style proceduralizer.
- 25 4. A system for automatic generation of card-based
presentation documents from multimedia data as claimed in
claim 3 wherein said style proceduralizer comprises:
a card-based context tree builder;
30 a content node path walker connected to said card-
based context tree builder; and
a content mapping rule generator connected to said
content node path walker.
- 35 5. A system for automatic generation of card-based
presentation documents from multimedia data as claimed in

claim 1 wherein said card based presentation generator comprises:

- a presentation construct mapper;
- a card-based DSSSL processor connected to said presentation construct mapper; and
- an FOD converter connected to said card-based DSSSL processor.

6. A system for automatic generation of card-based presentation documents from multimedia data comprising:
presentation style transformer means for receiving a card layout style specification and a card display schema and for providing a card-based presentation specification; and

card-based presentation generator means connected to said presentation style transformer means for receiving said card-based presentation specification and a card-based document content and for providing formatting object descriptions.

7. A system for automatic generation of card-based presentation documents from multimedia data as claimed in claim 6 wherein said presentation style transformer means comprises:

resource generator means for receiving said card layout style specification and said card display schema and for providing said card-based presentation specification; and

style proceduralizer means for receiving said card layout style specification and said card display schema and for providing said card-based presentation specification.

8. A system for automatic generation of card-based presentation documents from multimedia data as claimed in

claim 7 wherein said style proceduralizer means comprises:

card-based context tree builder means for receiving said card layout style specification and for providing a
5 context tree;

content node path walker means connected to said card-based context tree builder means for receiving said context tree and for providing context paths; and

content mapping rule generator means connected to
10 said content node path walker means for receiving said context paths and said card display schema and for providing said card-based presentation specification.

9. A system for automatic generation of card-based
15 presentation documents from multimedia data as claimed in claim 8 wherein said context tree captures content mapping rule context for making an efficient generation process of procedural rule mappings in CPS.

20 10. A system for automatic generation of card-based presentation documents from multimedia data as claimed in claim 6 wherein said card based presentation generator means comprises:

presentation construct mapper means for receiving
25 said card-based presentation specification and for providing card-based DSSSL style specifications;

card-based DSSSL processor means connected to said presentation construct mapper means for receiving said card-based DSSSL style specifications and said card-based
30 document content and for providing a card-based document flow object tree; and

FOD converter means connected to said card-based DSSSL processor means for receiving said card-based document flow object tree and for providing said
35 formatting object descriptions.

11. A system for automatic generation of card-based presentation documents from multimedia data as claimed in claim 10 wherein said card-based document flow object tree comprises:

a specification of a sequence of FODfo flow objects.

12. A method for automatic generation of card-based presentation documents from multimedia data comprising the steps of

transforming a presentation style; and
generating a card based presentation.

13. A method for automatic generation of card-based presentation documents from multimedia data as claimed in claim 12 wherein transforming a presentation style comprises the steps of:

resource generating presentation resource descriptions; and

translating declarative card layout style specifications into procedural card-based presentation specifications.

14. A method for automatic generation of card-based presentation documents from multimedia data as claimed in claim 13 wherein translating declarative card layout style specifications comprises the steps of:

building a card-based context tree;

building context paths; and

generating a content mapping rule.

15. A method for automatic generation of card-based presentation documents from multimedia data as claimed in claim 14 wherein generating a content mapping rule comprises the steps of:

getting a next context path;
deciding whether last path;
visiting a next node;
deciding whether end of path;
5 deciding whether node is visited before;
creating a context attribute value;
creating <DefineVAr>s for rule mapping;
creating <CaseExpr> or <IfExpr> if multiple node
types exist; and
10 deciding a node category.

16. A method for automatic generation of card-based
presentation documents from multimedia data as claimed in
15 claim 12 wherein generating a card based presentation
comprises the steps of:
mapping CPS constructs into card-based DSSSL style
constructs;
creating card-based document flow object tree; and
20 converting card-based document flow object tree into
formatting object descriptions.

Abstract of the Disclosure

5 An automatic card-based presentation generation
system programmatically creates card-based presentation
documents from source documents by transforming a card
layout style specification into presentation
specifications which at run-time, are used to generate
10 formatting object descriptions. The formatting object
descriptions are then used to actually create card-based
presentation documents. The invention includes a
presentation style transformer and a card-based
presentation generator. The presentation style
15 transformer takes a card layout style specification
(CLSS) and a card display schema (CDS) and transforms
them into card-based presentation specifications (CPS).
The card-based presentation generator takes the card-
based presentation specifications and a card-based
20 document content to create formatting object
descriptions.

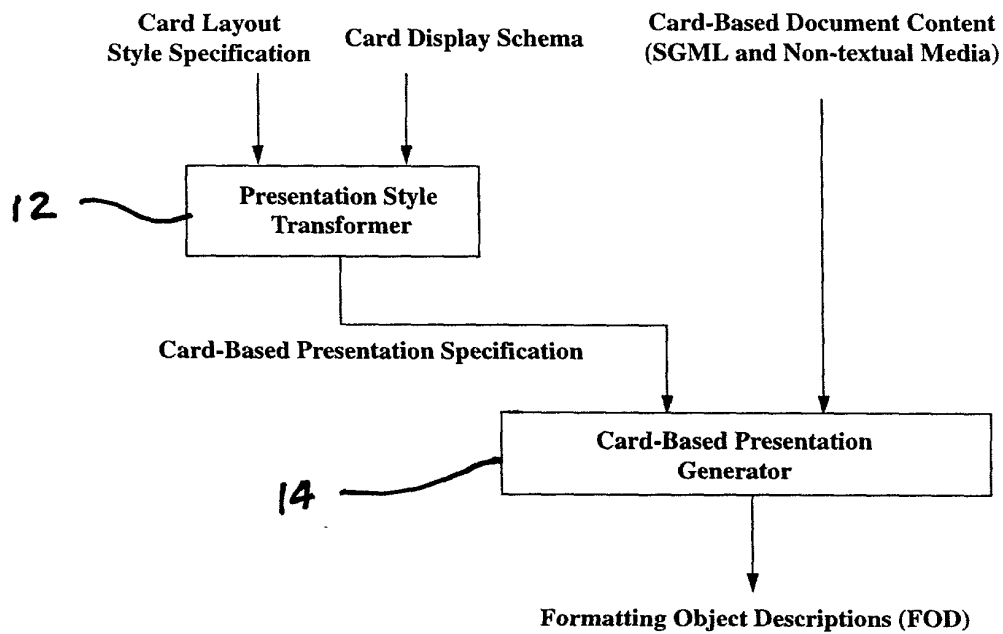


Figure 1. Presentation Generation Process

```

<CardLayoutStyle Name=name ... >
  <Options size=card_size ... > ... </Options>
  ...
  <BackgroundList>
    <Background name=name Func=function_name>
      <object property=... ></>
    ...
    </Background>
    <Background name=name Func=function_name>
      <object property=... ></>
    ...
    </Background>
  ...
</BackgroundList> ...
  <ForegroundList>
    <Foreground name=name Func=function_name>
      <object property=... ></>
    ...
    </Foreground>
    <Foreground name=name Func=function_name>
      <object property=... ></>
    ...
    </Foreground>
  ...
</ForegroundList>
  <CardSeq name=name Context=context Type=type Content=content>
    <Background StyleRef=background_name>
  </CardSeq>
  <CardSeq name=name Context=context Type=type Content=content>
    <Background StyleRef=background_name>
  </CardSeq>
  ...
  <Card name=name Context=context Type=type Content=content>
    <Foreground StyleRef=background_name></>
  </Card>
  <Card name=name Context=context Type=type Content=content>
    <Foreground StyleRef=background_name></>
  </Card>
  ...
</CardLayoutStyle>

```

Figure 2. Card Layout Style Specification

```

<CardLayoutStyle Name=name . . .>
  <Options size=card_size . . . > . . . </Options>
  . . .
  <BackgroundList>
    <Background name=name Func=function_name>
      <object property= . . . ></>
      . . .
    </Background>
    <Background name=name Func=function_name>
      <object property= . . . ></>
      . . .
    </Background>
    . . .
  </BackgroundList> . . .
  <ForegroundList>
    <Foreground name="TextByGfxForeground" Func="TextByGfxForeground">
      <Object Name="TextByGfxTitle"
        Coords="180,1395,1150,1305"
        Type="Field"
        Func="TitleText"
        Context="Card,CardX, CardTitle"
        FontSize="15"
        FontStyle="Bold"
        FontColor="blue"
        Content="CardTitle"></>

      <Object Name="GfxContent"
        Coords="357 94 881 693"
        Type="Graphics"
        Func="GfxContent"
        Context="Card,CardX, AnyDocX, Figure, Graphic"
        Content="Graphic"
        AIU="Card, CardX, AnyDocX, Figure, AIUDoc, ROI"></>
      . . .
    </Foreground>
    . . .
  </ForegroundList>
  <CardSeq name="TextByGfxSeq Type="TextByGfx"
    Context="PlantLevel,CardSeq" Content="">
    <Background StyleRef="NormalBackground">
  </CardSeq>
  . . .
  <Card name="TextByGfxCard" Type="TextByGfx"
    Context="PlantLevel, CardSeq, Card" Content="">
    <Foreground StyleRef="TextByGfxForeground"></>
  </Card>
  . . .
</CardLayoutStyle>

```

Figure 3. Example of CardTitle Layout Style Specifications

```

<MACRO_Declaration>
  <ObjectGroup name="project_resource">
    MACRO_Resource(stylename, type, filename)</>
    MACRO_Resource(stylename, type, filename)</>
    ...
  </ObjectGroup>
  MACRO_name(stylename, parameters) </>
  MACRO_name(stylename, parameters) </>
  ...
</MACRO_Declaration>
<ContentMapping>
  <ElementRule Context=doc_tags>
    <DefineVar name=...>
    <DefineVar name=...>
    ...
    <CaseExpr key= ... >
    <CaseClause matchvalue= ... >
    <AddObject otype= ... stylename= ... >
    <ProcessGrove command= ... ></>
    ...
    <AddGroup groupname= ... ></>
    ...
  </AddObject>
  </CaseClause>

  <CaseClause matchvalue= ... >
  <AddObject otype= ... stylename= ... >
  <ProcessGrove command= ... ></>
  ...
  <AddGroup groupname= ... ></>
  ...
  </AddObject>
  </CaseClause>
  </CaseExpr>
  </ElementRule>

  <ElementRule ... > ... </>
  ...
</ContentMapping>

```

Figure 4. Card-Based Presentation Specification

<MACRO_Declaration>

...

...

MACRO_TitleField(**TextByGfxTitle**, "180,1395,1150, 1305, "\$CardTitle_RTF\$, 15", "bold", "blue")

...

</MACRO_Declaration>

<ContentMapping>

...

<ElementRule context="(Card CardX)">

<ProcessGrove

command="process-matching-children **CardTitle**">

<ProcessGrove

command="process-matching-children **ANYDOCX**">

</ElementRule>

<ElementRule context="(Card CardX CardTitle)">

<DefineVar name="\$CardTitle_RTF" forvalue="(RTF-TitleGen (current node))"></>

<CaseExpr key= "(CardType (current-node))">

<CaseClause matchvalue= "GfxonlyCard">

<AddObject otype="Field" stylename="GfxOnlyTitle"></>

</CaseClause>

<CaseClause matchvalue= "TextByGfxtitle">

<AddObject otype="Field" stylename="**TextByGfxTitle**"></>

</CaseClause>

<CaseClause matchvalue= "else">

<AddObject otype="Field" stylename="TextByGfxTitle"></>

</CaseClasuse>

</CaseExpr>

</ElementRule>

<ElementRule> ... </>

...

</ContentMapping>

Figure 5. Example of Card-Based Presentation Specification for CardTitle

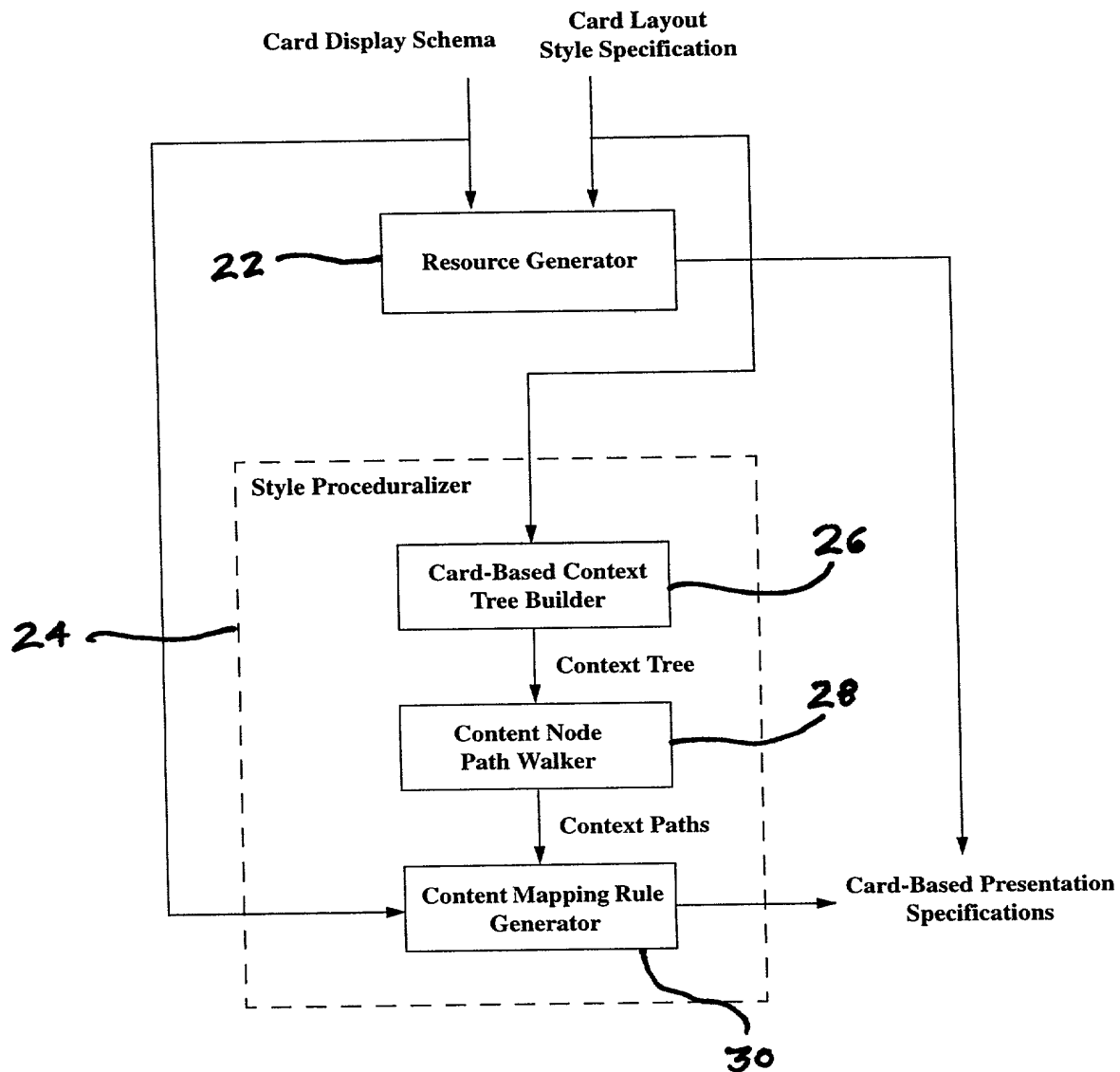


Figure 6. Presentation Style Transformer

```

<DisplaySchema Objecttype="Field" Name="TextByGfxTitle" >
  <Argument name="StyleName" type="Required"
    AvailVule="#ANY_UNIQUE_NAME">
  <Argument name="vertices" type="Required"
    Availalue="#ANY_2POINTS">
  <Argument name="TitleRTF" type="Optional"
    DefaultValue=$CARDTitle_RTF$ >
  <Argument name="FontSize" type="Optional"
    DefaultValue="15" AvailValue="#Number">
  <Argument name="Fontstyle" type="Optional"
    DefaultValue="Regular" AvailValue="Bold,Regular, Italic">
  <Argument name="FontColor" type="Optional"
    DefaultValue="Black" AvailValue="#ANY_Color_Name or RGBStoke">
</DisplaySchema>

```

Figure 7. Display Schema for Generating MACRO_ TitleField

```

<DisplaySchema Objecttype= object_type Name= macro_name or variable_name>
  <Argument name= . . . type= . . .
    DefaultValue= . . . AvailValue= . . . >
  <Argument name= . . . type= . . .
    DefaultValue= . . . AvailValue= . . . >
  <Argument name= . . . type= . . .
    DefaultValue= . . . AvailValue= . . . >
  . . .
</DisplaySchema>

```

Figure 8. Display Schema for Generating Resource Declarations or Variable Definition


```

<DisplaySchema Objecttype="Resource" MacroName="MACRO_Resource">
  <Argument name="Format" type="Optional"
    DefaultValue="icon" AvailValue="icon,bitmap,sharedscript, menubar">
  <Argument name="StyleName" type="Required"
    AvailVule="#ANY_UNIQUE_NAME">
  <Argument name="InstanceName" type="Optional"
    DefaultValue="" AvailValue="#ANY_STRING">
  <Argument name="FileName" type="Required"
    AvailValue="#ANY_STRINGr">
</DisplaySchema>

```

Figure 10. Display Schema for Generating all Primitive MACRO_ Resources

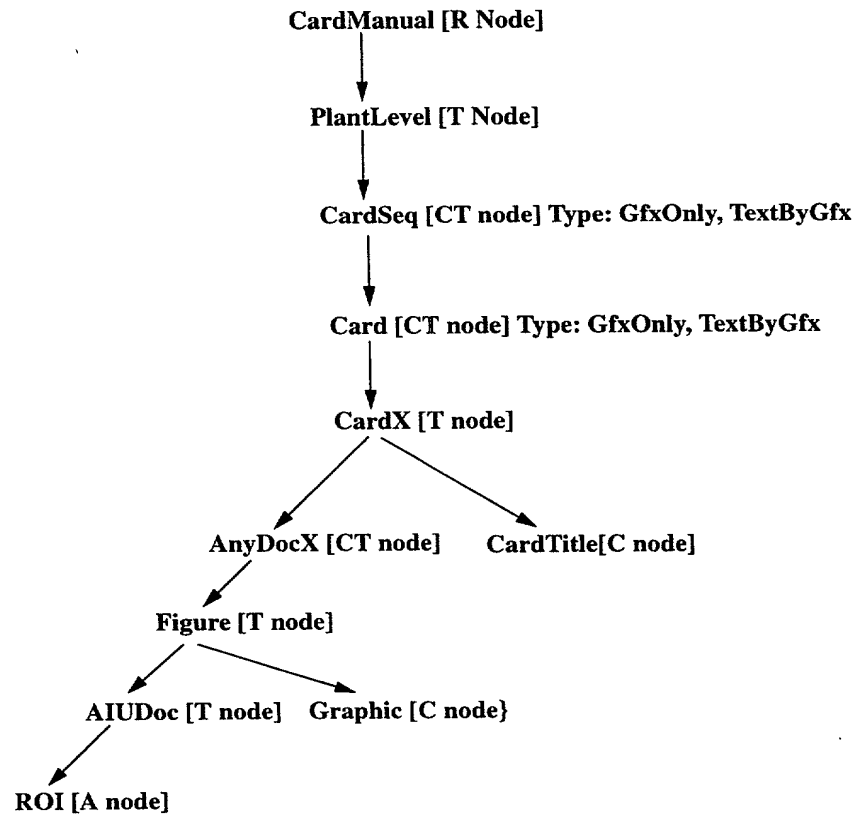


Figure 11. Example of Card-Based Context Tree

1. CardManual, PlantLevel, CardSeq
2. CardManual, PlantLevel, CardSeq, Card
3. CardManual, PlantLevel, CardSeq, Card, CardX, CardTitle
4. CardManual, PlantLevel, CardSeq, Card, CardX, AnyDocX
5. CardManual, PlantLevel, CardSeq, Card, CardX, AnyDocX, Figure, Graphic
6. CardManual, PlantLevel, CardSeq, Card, CardX, AnyDocX, Figure, AIUDoc, ROI

Figure 12. Example of Card-Based Context Paths for C, CT and A Nodes

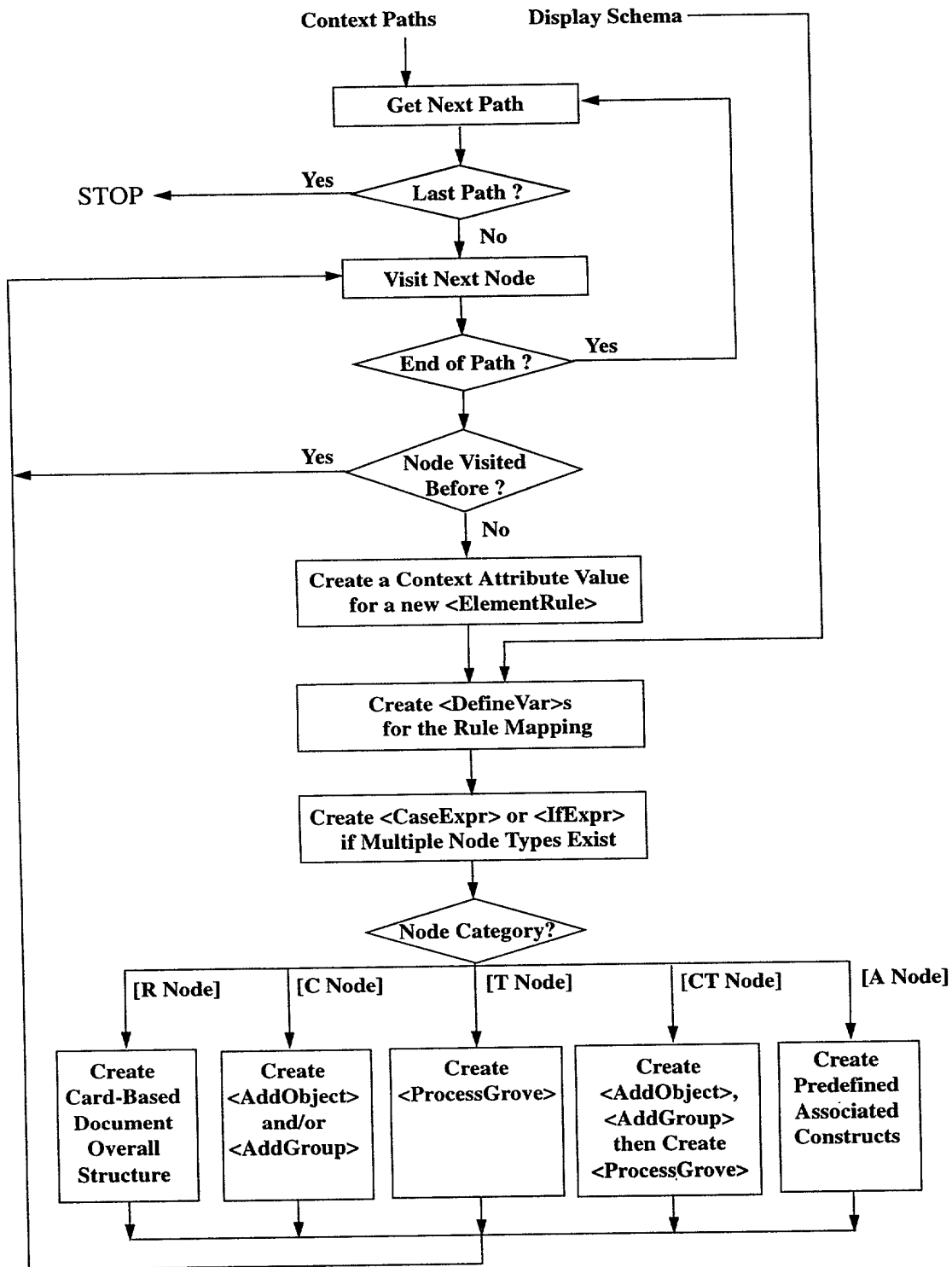


Figure 13. Content Mapping Rule Generation Process

(A) Transition Node: CardX

```
<ElementRule context="(Card CardX)">
  <ProcessGrove
    command="process-matching-children CARDTITLE"></>
  <ProcessGrove
    command="process-matching-children ANYDOCX"></>
</ElementRule>
```

(B) Content Node: CardTitle (See Figure 5)

(C) Content Transition Node: CardSeq

```
<ElementRule context="(PlantLevel CardSeq)">
  <DefineVar name="$Page_currentObject$"
    strValue="(currentObjectGen (current-node))"> </>
  <DefineVar ...> </>
  ...
  <CaseExpr key="(cardSeqType (current-node))">

    <CaseClause matchval="GfxOnly">
      <AddObject otype=Background styleName="GfxOnlyBackground">
        <ProcessGrove
          command="process-matching-children CARD"></>
        </AddObject>
      </CaseClause>

      <CaseClause matchval="TextByGfx">
        <AddObject otype=Background styleName="NormalBackground">
          <AddObject otype=Button styleName="Top"></>
          <AddObject otype=Button styleName="First"></>
          <AddObject otype=Button styleName="Previous"></>
          <AddObject otype=Button styleName="Next"></>
          <AddObject otype=Button styleName="Last"></>
          <AddObject otype=Button styleName="Audio"></>
          <AddObject otype=Button styleName="Animate"></>
          ...
        <ProcessGrove
          command="process-matching-children CARD"></>
        </CaseClause>

        <CaseClause matchval="else">
          ...
        </CaseClause>

      </CaseExpr>
    </ElementRule>
```

Figure 14. Example of Generated CPS for C, CT and T Nodes

(D) ROOT NODE: CardManual

```
<ElementRule context="CardManual">
  <AddObject otype=CardManualLayout styleName="">
    <DefineVar name="$BookTOC$" strValue="(TOC-Gen (current-node))"> </>
    <AddObject otype=ToolBookManual styleName="IMIS_ToolBook">
      <AddGroup groupName="Project _Resources"> </>
      <AddObject otype=MainViewer styleName="MainViewer">
        <ProcessGrove
          command="process-matching-children PLANTLEVEL"></>
        <AddGroup groupName="IMIS_MiscViewersContent"> </>
      </AddObject>
      <AddGroup groupName="IMIS_MiscViewers"> </>
    </AddObject>
  </AddObject>
</ElementRule>
```

(E) Associated Node: ROI

```
<ElementRule context="(Card CardX AnyDocX Figure AIUDoc ROI)">
  <DefineVar name="$AiuName$"
    strValue="(getattr NAME (current-node))"> </>

  <CaseExpr key="(cardType (current-node))">

    <CaseClause matchval="GfxOnly"> ...</CaseClause>
    <CaseClause matchval="TextByGfx">
      <CaseExpr key="(ROI-TYPE (current-node))">
        <CaseClause matchval="polygon">
          <AddObject otype=Polygon styleName="GfxContent-AIU-Poly"></>
        </>
        <CaseClause matchval="callout"> ..</>
        <CaseClause matchval="NamedCallout"> ..</>
        ...
      </CaseExpr>
    </CaseClause>
    <CaseClause matchval="else">
      <EmptySosofo>
    </CaseClause>

  </CaseExpr>
</ElementRule>
```

Figure 15. Example of Generated CPS for R and A Nodes

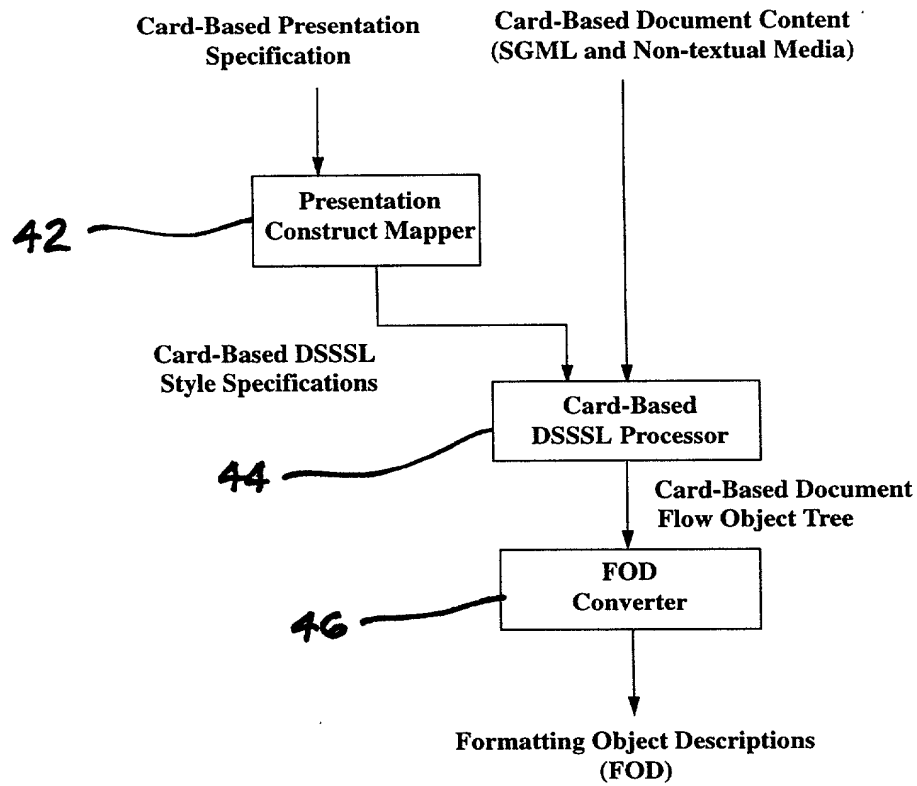


Figure 16. Hypermedia Document Presentation Generator

```

(element (Card CardX CardTitle)
  (sosof-append
    (makeFODfo FOfotype: "DefineVar"
      FOfname: "$CardTitle_RTF$"
      FOinstruction: (RTF-TitleGen (current Node)))
    (case (cardType (current-node))
      (("GfxOnly")
        (makeFODfo FOfotype: "StartTag"
          FOfname: "Field"
          FOinstruction: (AttributeGen GfxOnlyTitle)))
      ("TextByGfx")
        (makeFODfo FOfotype: "StartTag"
          FOfname: "Field"
          FOinstruction: (AttributeGen TextByGfxTitle)))
      (else
        (makeFODfo FOfotype: "StartTag"
          FOfname: "Field"
          FOinstruction: (AttributeGen TextByGfxTitle)))
      )
    )
  )
)

```

Figure 17. Generated Card-Based DSSSL Style for CardTitle

```

(define (AttributeGen type)
(case type
(("TextByGfxTitle")
(string-append
  " FontSize=\"15\" "
  " FontStyle=\"bold\" "
  " FontColor=\"0,0,255\" "
  " RGBFILL=\"255,255,255\" "
  " TRANSPARENT=\"true\" "
  " VERTICES=\"180,1395,1150,1305\" "
  " RICHTEXT=\"$CardTitle_RTF$\" "
  " FONTFACE=\"Times New Roman\" "
  " ACTIVATED=\"true\" "
  " BASELINES=\"false\" "
  " BORDERSTYLE=\"none\" "
  " FIELDTYPE=\"wordWrap\" "
  " SCROLL=\"0\" "
  " SPACING=\"1\" "
  " INDENTS=\"0,0,0\" "
  " TABSPACING=\"360\" "
  " TABTYPE=\"left\" "
  " TEXTALIGNMENT=\"left\" "
  " NAME=\"TitleField\""))
(("GfxOnlyTitle")
...
...
)

(define (makeFODfo #! key FOType FOname FOinstruction)
(case FOType
(("StartTag")
(make FOD-Instruction
  data: (string-append "<" FOname FOinstruction "> "</>" )))
(("DefineVar")
(make FOD-Instruction
  data: (string-append "<DefineVar name=" FOname "value=" FOinstruction ">")))
(("EndTag")
(make FOD-instruction
  data: (string-append "</>")))
...
(else
(make FOD-instruction
  data: (string-append "newFlowObjectType"))))
)

```

Figure 18. Card-Based DSSSL Style Functions for CardTitle

```

<CardFormattingBook property=...>
  <Resource property=...>...</Resource>
  <Resource property=...>...</Resource>
  ...
  <MainViewer property=...>
    <StartBkgnd property=...>
      <Object property=...>
        ...
        <StartPage property=...>...</StartPage>
        <NextPage property=...>
          <Field Name="TitleField"
            TextAligment="left"
            TabType="left"
            TabSpacing="360"
            Scroll="0"
            FieldType="wordwrap"
            BorderStyle="none"
            BaseLines"false"
            Activated="true">
              Fontface="Time New Roman"
              Vertice="180,1395,1150,1305"
              Transparent="True"
              RichText=
                "{\rtf1\ansi \deff5\deftab360\deflang1033
                {\fonttbl
                {\f1\froman\fcharset2\fs20 Symbol;}
                {\f2\fttech Marlett;}
                {\f4\froman\fcharset0\fs20 Times New Roman;}
                {\f5\fswiss\fcharset0\fs20 Arial;}
                {\f10\fswiss\fcharset0\fs20 System;}}
                {\colortbl;\red0\green0\blue0;\red0\green0\blue255;
                \red255\green0\blue0;\red255\green255\blue255;}
                \pard\plain \f4\fs20 {\b\i\fs36\cf1 CO2 Purge System 1}
                }"
              RGBFill="255,255,255"
              fontsize="15"
              fontStyle="bold"
              rgbStroke="0, 0, 255"
            >
          </NextPage>
          <NextPage property=...>...</NextPage>
          ...
          </StartBkgnd>
          <NextBkgnd property=...>...</NextBkgnd>
          ...
        </MainViewer>
        <Viewer property=...>...</Viewer>
        ...
      </CardFormattingBook>

```

Figure 19. CardTile Formatting Object Description

```

<CardFormattingBook property=...>
  <Resource property=...>... </Resource>
  <Resource property=...>... </Resource>
  ...
  <MainViewer property=...>
    <StartBkgnd property=...>
      <Object property=...>
        ...
        <StartPage property=...>
          <Object property=...>
            ...
            </StartPage>
          <NextPage property=...>
            <Object property=...>
              ...
              </NextPage>
            <NextPage property=...>
              <Object property=...>
                ...
                </NextPage>
              ...
            </StartBkgnd>
          <NextBkgnd property=...>... </NextBkgnd>
          ...
        </MainViewer>
        <Viewer property=...>... </Viewer>
        <Viewer property=...>... </Viewer>
        ...
      </CardFormattingBook>

```

Figure 20. Card-Based Formatting Object Description